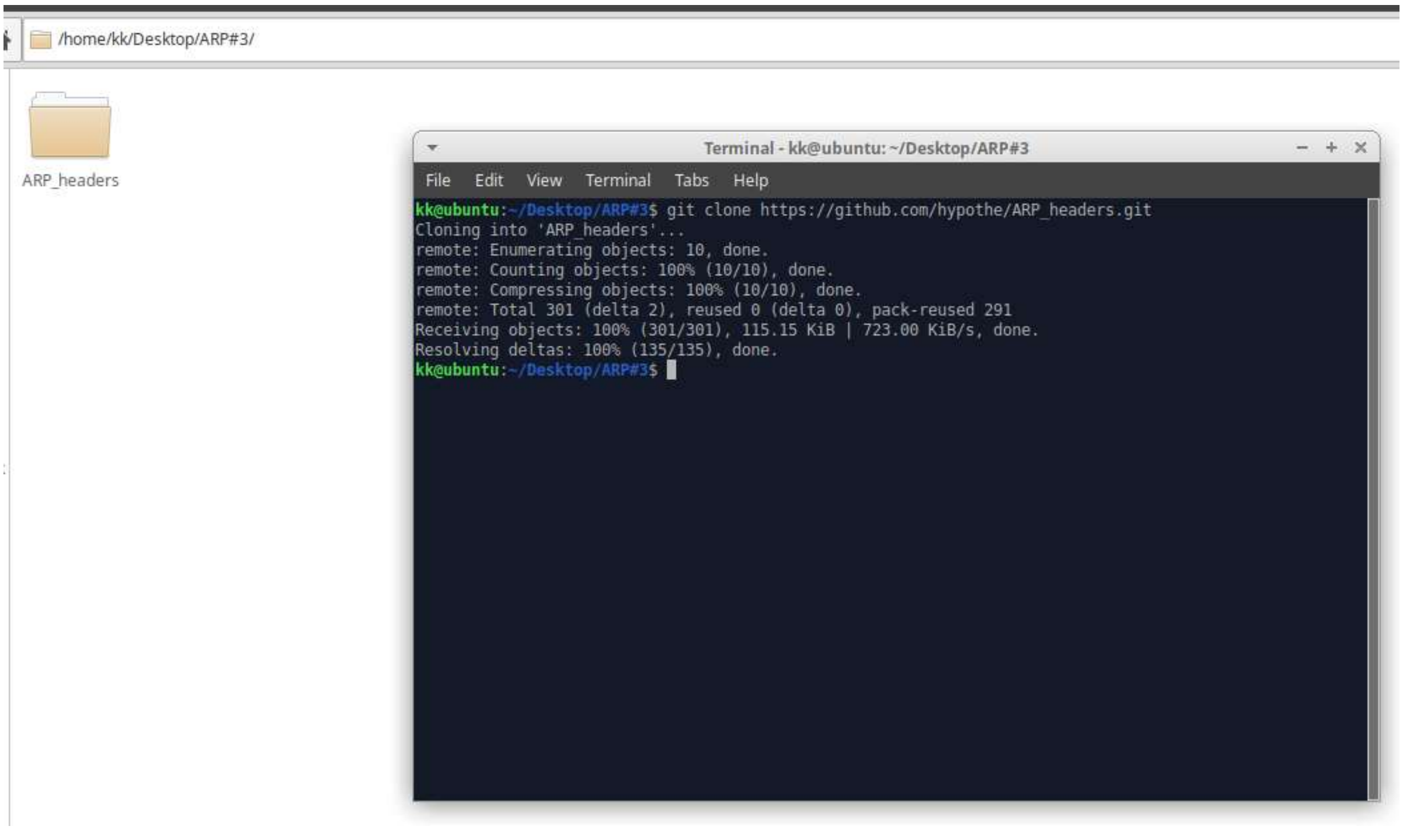
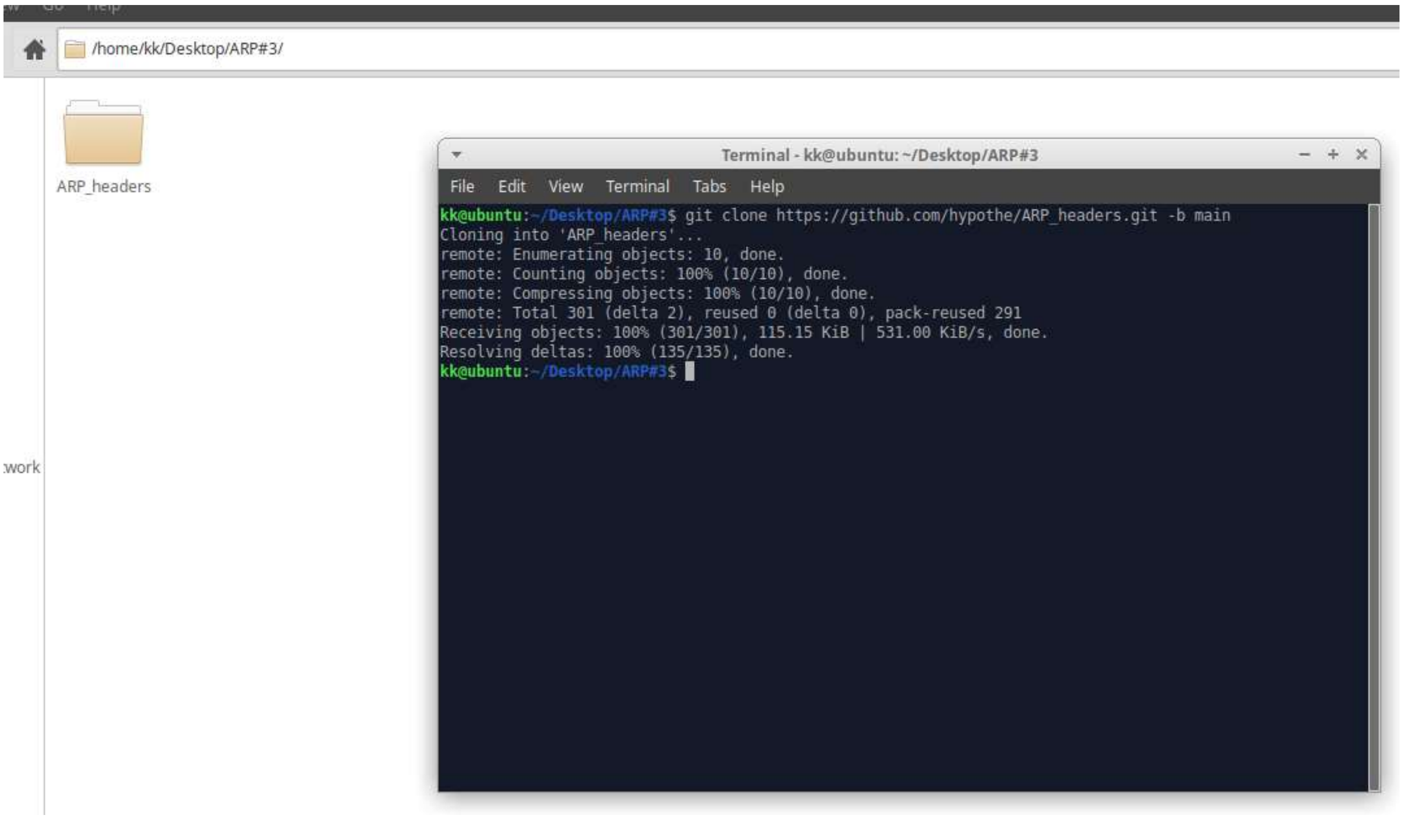
ARP#3 – HOW TO BUILD YOUR PROJECT - using ARP shared library

Step 1: download the shared library; a folder *ARP_headers* will be created.


/home/kk/Desktop/ARP#3/
ARP_headers
Terminal - kk@ubuntu: ~/Desktop/ARP#3
File Edit View Terminal Tabs Help

```
kk@ubuntu:~/Desktop/ARP#3$ git clone https://github.com/hypothe/ARP_headers.git
Cloning into 'ARP_headers'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 301 (delta 2), reused 0 (delta 0), pack-reused 291
Receiving objects: 100% (301/301), 115.15 KiB | 723.00 KiB/s, done.
Resolving deltas: 100% (135/135), done.
kk@ubuntu:~/Desktop/ARP#3$
```

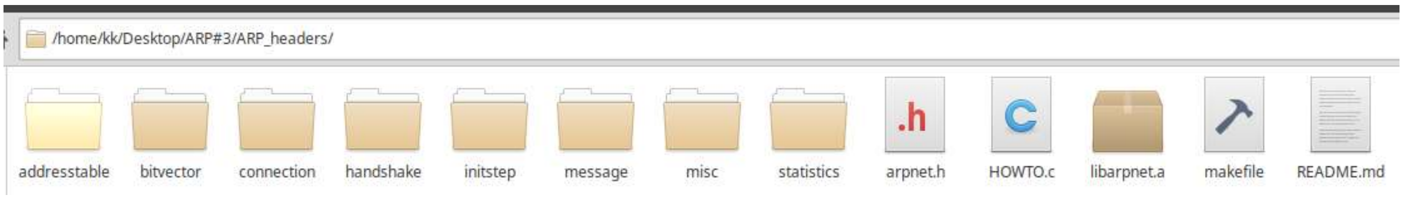
I suggest you choose precisely the branch *main*, attaching at the end of the command *-b main* .


/home/kk/Desktop/ARP#3/
ARP_headers
work
Terminal - kk@ubuntu: ~/Desktop/ARP#3
File Edit View Terminal Tabs Help

```
kk@ubuntu:~/Desktop/ARP#3$ git clone https://github.com/hypothe/ARP_headers.git -b main
Cloning into 'ARP_headers'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 301 (delta 2), reused 0 (delta 0), pack-reused 291
Receiving objects: 100% (301/301), 115.15 KiB | 531.00 KiB/s, done.
Resolving deltas: 100% (135/135), done.
kk@ubuntu:~/Desktop/ARP#3$
```

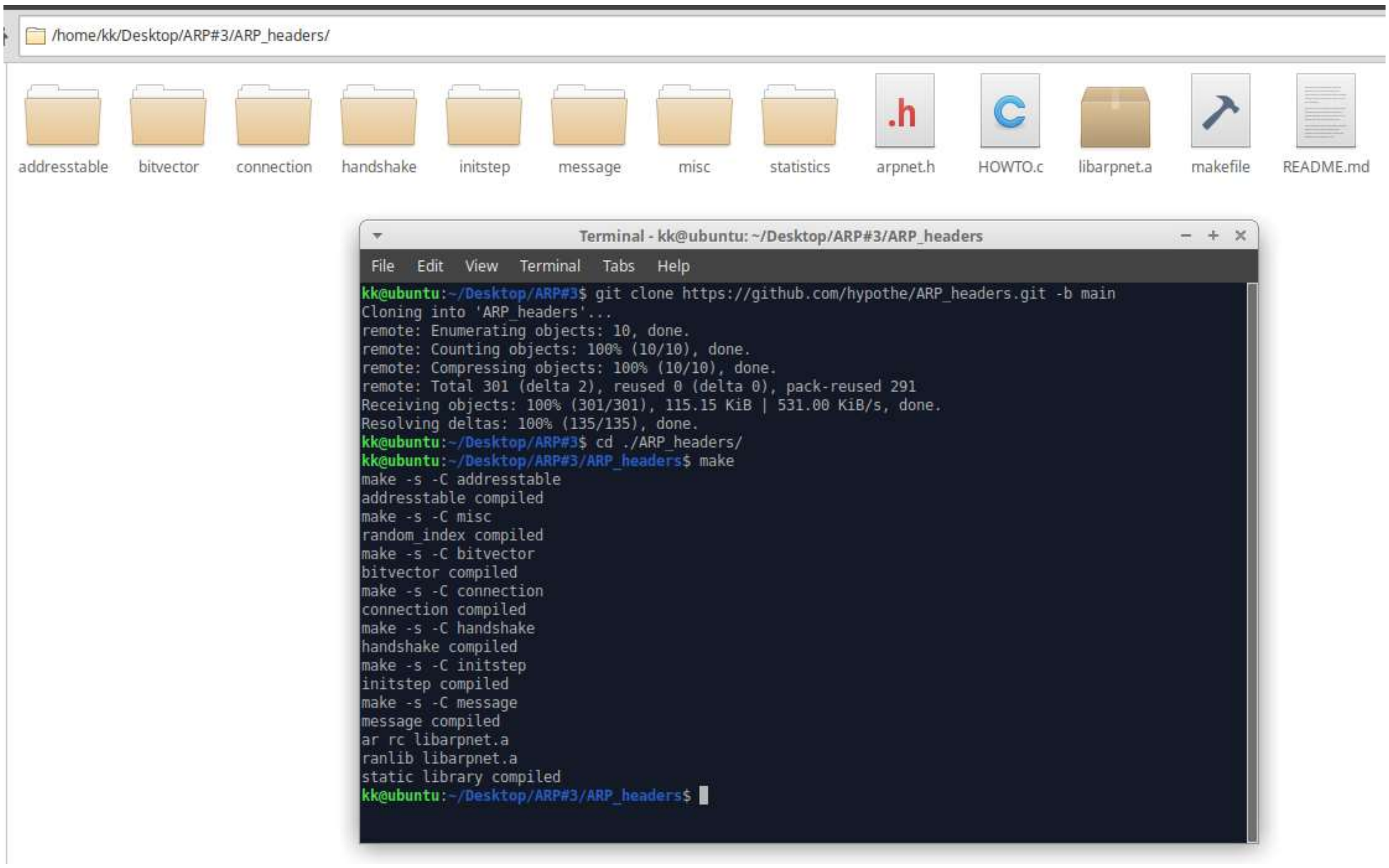
Step 2: open the folder *ARP_headers*.

In order to compile the library you must have a file within it named *libarpnet.a* .



This file couldn't exist, so you have to build it.

Launch a terminal here, and write *make*:

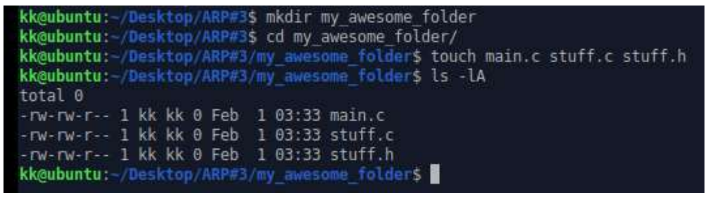


```
kk@ubuntu:~/Desktop/ARP#3$ git clone https://github.com/hypothe/ARP_headers.git -b main
Cloning into 'ARP_headers'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 301 (delta 2), reused 0 (delta 0), pack-reused 291
Receiving objects: 100% (301/301), 115.15 KiB | 531.00 KiB/s, done.
Resolving deltas: 100% (135/135), done.
kk@ubuntu:~/Desktop/ARP#3$ cd ./ARP_headers/
kk@ubuntu:~/Desktop/ARP#3/ARP_headers$ make
make -s -C addresstable
addresstable compiled
make -s -C misc
random_index compiled
make -s -C bitvector
bitvector compiled
make -s -C connection
connection compiled
make -s -C handshake
handshake compiled
make -s -C initstep
initstep compiled
make -s -C message
message compiled
ar rc libarpnet.a
ranlib libarpnet.a
static library compiled
kk@ubuntu:~/Desktop/ARP#3/ARP_headers$ 
```

This file is necessary to compile your code because the library have to be *linked* to your object files.

Step 3: set up your environment and code

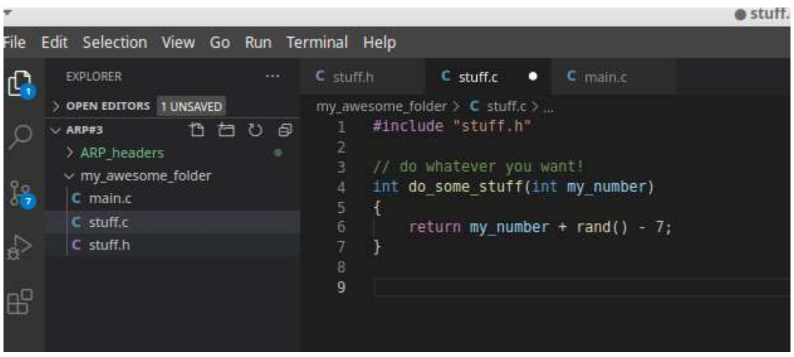
Let's say we have three source files in a folder outside ARP_headers; all my code is contained in a folder called *my_awesome_folder* immediately outside the one which contains the library. Let's create it:



```
kk@ubuntu:~/Desktop/ARP#3$ mkdir my_awesome_folder
kk@ubuntu:~/Desktop/ARP#3$ cd my_awesome_folder/
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ touch main.c stuff.c stuff.h
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ ls -lA
total 0
-rw-rw-r-- 1 kk kk 0 Feb  1 03:33 main.c
-rw-rw-r-- 1 kk kk 0 Feb  1 03:33 stuff.c
-rw-rw-r-- 1 kk kk 0 Feb  1 03:33 stuff.h
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$
```

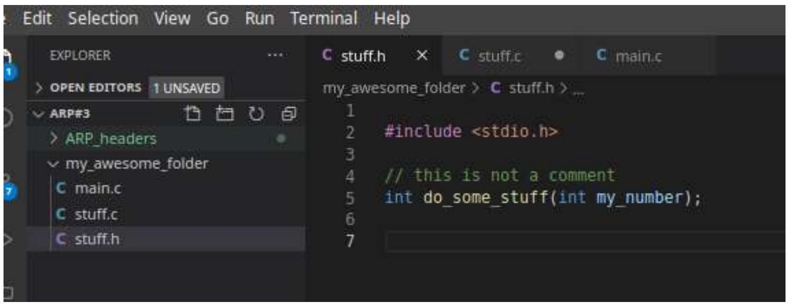
Now, you can coding within this folder. Let's say, we wrote this:

stuff.c



```c
#include "stuff.h"

// do whatever you want!
int do_some_stuff(int my_number)
{
    return my_number + rand() - 7;
}

```

stuff.h

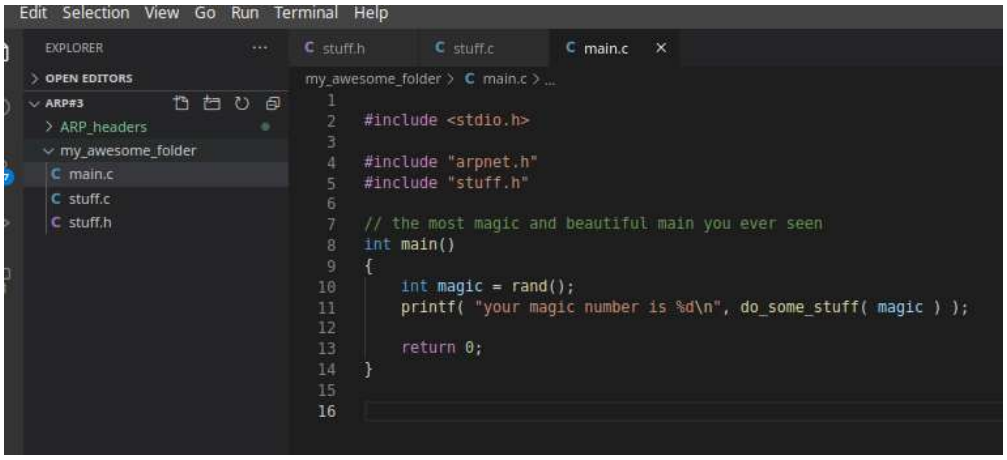


```c

#include <stdio.h>

// this is not a comment
int do_some_stuff(int my_number);

```

For using the ARP shared library, you must simply add

#include “arpnet.h”

As I'm showing there:



```c

#include <stdio.h>

#include "arpnet.h"
#include "stuff.h"

// the most magic and beautiful main you ever seen
int main()
{
    int magic = rand();
    printf( "your magic number is %d\n", do_some_stuff( magic ) );

    return 0;
}

```

Step 4: time to compile!

Before compiling, look at this instruction (you can find this in the readme inside the library):

```
gcc -L<path>/ARP_headers -I<path>/ARP_headers -o <exe_name> <list_source_names>.c -larpnet
```

in which

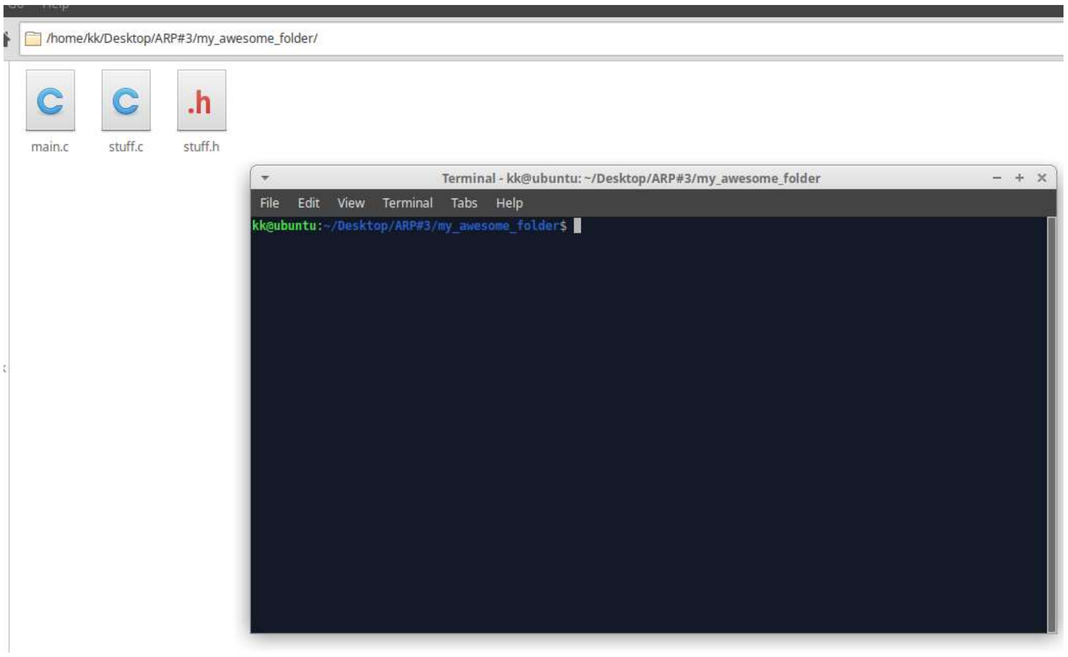
- **path** is the folder containing all the code of the shared library, *with respect to your workspace*: this means that if you're compiling from the folder I showed to you, the path is simply point-point, ..
- **exe name** is the name of the executable you have at the end of the compilation
- **list_source_names** is the list of all the object files of your program; in our case, *main.o stuff.o*

the final part of the instruction, *-larpnet*, performs the linking with the library.

In our example, because our program is composed by two files, we have first of all to produce the object files, and then compile them using the command

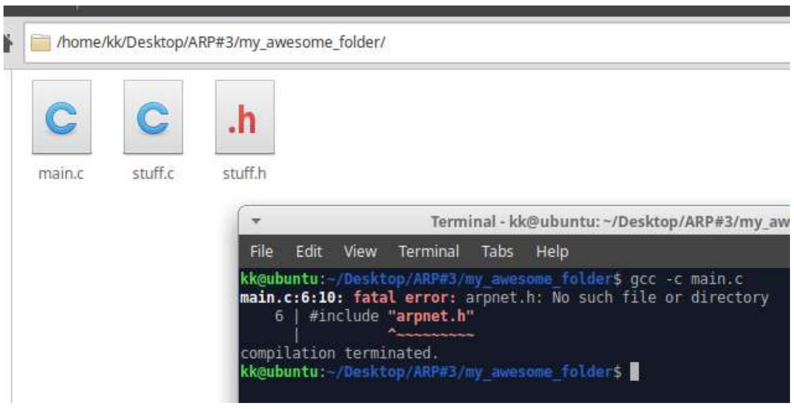
```
gcc -L../ARP_headers -I../ARP_headers -o my_executable main.o stuff.o -larpnet
```

Remember that you can write a simple script for automatizing this operation. Let's build! Launch a terminal in your workspace:


/home/kk/Desktop/ARP#3/my_awesome_folder/
main.c
stuff.c
stuff.h
Terminal - kk@ubuntu: ~/Desktop/ARP#3/my_awesome_folder
File Edit View Terminal Tabs Help

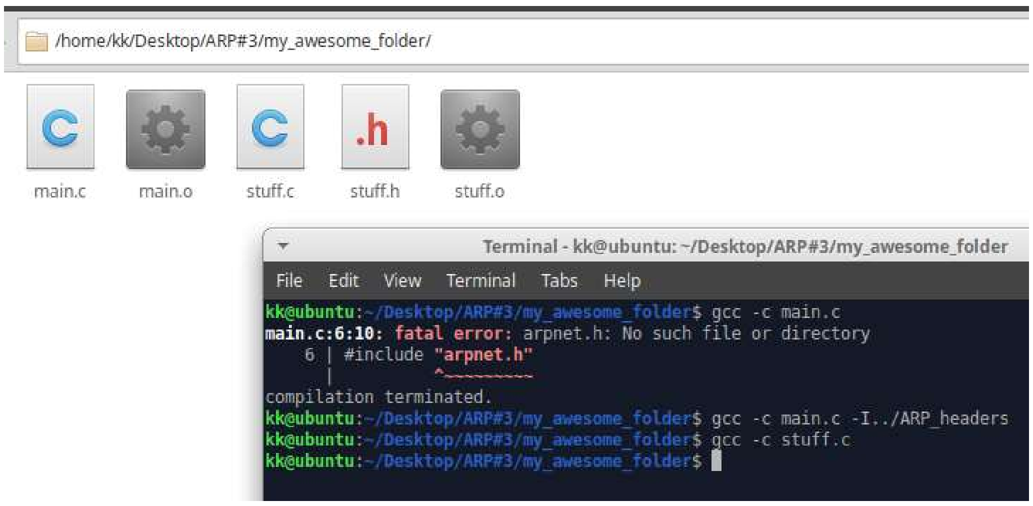
```
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$
```

Build the object files… pitfall you may face:


/home/kk/Desktop/ARP#3/my_awesome_folder/
main.c
stuff.c
stuff.h
Terminal - kk@ubuntu: ~/Desktop/ARP#3/my_aw
File Edit View Terminal Tabs Help

```
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ gcc -c main.c
main.c:6:10: fatal error: arpnet.h: No such file or directory
    6 | #include "arpnet.h"
      |          ^~~~~~~~~~
compilation terminated.
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$
```

This happens because the compiler cannot find the header inside your workspace … trivial: it is *outside* it. Use the option *-I../ARP_headers* :

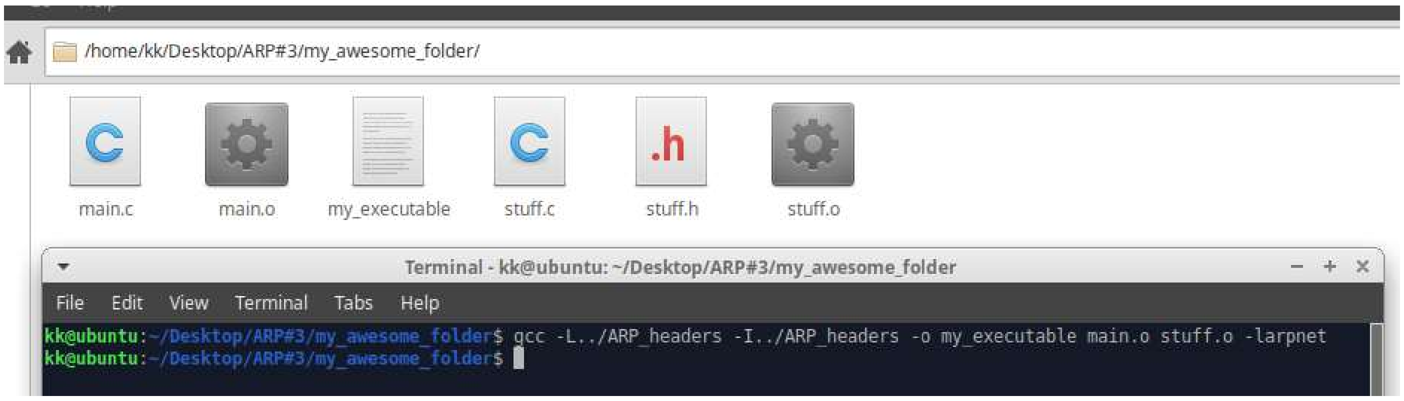

/home/kk/Desktop/ARP#3/my_awesome_folder/
main.c
main.o
stuff.c
stuff.h
stuff.o
Terminal - kk@ubuntu: ~/Desktop/ARP#3/my_awesome_folder
File Edit View Terminal Tabs Help

```
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ gcc -c main.c
main.c:6:10: fatal error: arpnet.h: No such file or directory
    6 | #include "arpnet.h"
      |          ^~~~~~~~~~
compilation terminated.
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ gcc -c main.c -I../ARP_headers
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ gcc -c stuff.c
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$
```

It works!

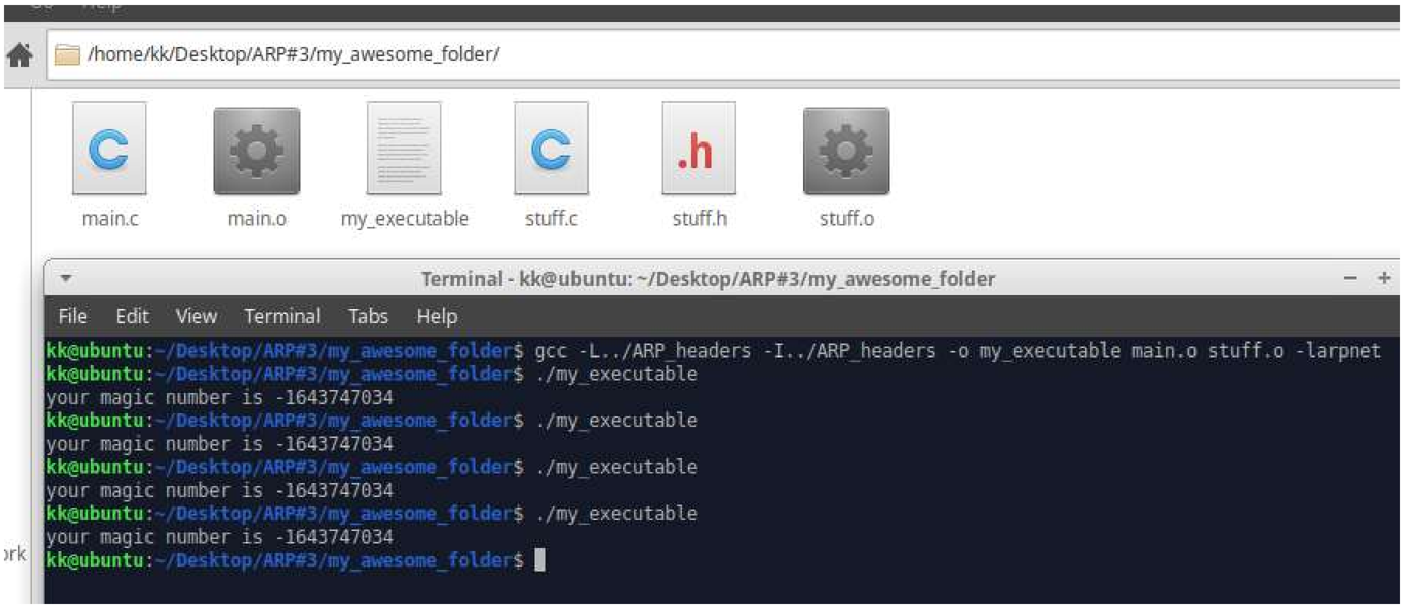
Note that *stuff.c* is compiled with no error; this because it doesn't directly import the header *arpnet.h*, so the compiler doesn't need to search for this file.

We're ready for the final compilation, use the above incantation:


/home/kk/Desktop/ARP#3/my_awesome_folder/
main.c
main.o
my_executable
stuff.c
stuff.h
stuff.o
Terminal - kk@ubuntu: ~/Desktop/ARP#3/my_awesome_folder
File Edit View Terminal Tabs Help

```
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ gcc -L../ARP_headers -I../ARP_headers -o my_executable main.o stuff.o -larpnet
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$
```

The code is now compiled! Let's test it:


/home/kk/Desktop/ARP#3/my_awesome_folder/
main.c
main.o
my_executable
stuff.c
stuff.h
stuff.o
Terminal - kk@ubuntu: ~/Desktop/ARP#3/my_awesome_folder
File Edit View Terminal Tabs Help

```
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ gcc -L../ARP_headers -I../ARP_headers -o my_executable main.o stuff.o -larpnet
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ ./my_executable
your magic number is -1643747034
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ ./my_executable
your magic number is -1643747034
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ ./my_executable
your magic number is -1643747034
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$ ./my_executable
your magic number is -1643747034
kk@ubuntu:~/Desktop/ARP#3/my_awesome_folder$
```

Let's say: it "*works*". That's all!